Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management,
May 14-18, 2006, New Orleans, Louisiana, http://www.psam8.org/ Paper PSAM-0193

INTERTWINING RISK INSIGHTS AND DESIGN DECISIONS

Steven L. Cornford
Jet Propulsion Laboratory,
California Institute of Technology

Martin S. Feather
Jet Propulsion Laboratory,
California Institute of Technology

J. Steven Jenkins
Jet Propulsion Laboratory,
California Institute of Technology

SUMMARY/ABSTRACT

The role of risk assessment in design is to yield insights that influence decisions. If done only at the *culmination* of the design process, the space of remaining options among which to decide is severely constrained. In response to late-lifecycle risk insights, changes to the designed system will be limited to fine tuning and modest refinements, with the only significant areas of variability remaining in the way that system is operated, maintained and (ultimately) decommissioned. These latter are accomplished by changes to not the system itself (which must be used more or less "as is"), but to the operational procedures, maintenance practices, and scenarios of use. Conversely, if risk assessment is done *early* and continued *throughout* the design process, opportunities exist to use the risk insights to influence both the design itself and how it is to be realized. Such early insights enable significant design changes before large and irrecoverable investments have been made.

The state of systems engineering is such that a form of early and continued use of risk assessments is conducted (as evidenced by NASA's adoption and use of the "Continuous Risk Management" paradigm developed by SEI). In recognition of inevitable future uncertainties as the design process unfolds, systems engineering practices include the establishment and tracking of pre-determined allocations of reserves of the kinds of resources seen to be critical to the design at hand (e.g., schedule, budget, mass, power). Risk assessment can be used to look ahead at the development plan and operational scenarios to identify significant risks. These risks can then be assessed in terms of their likelihoods, their potential impacts on the critical resources (e.g., cost, schedule and functionality), and the options for preventing/reducing risks or for workarounds should they occur. However, these practices fall short of the ideal: (1) Integration between risk assessment techniques and other systems engineering tools is weak. (2) Risk assessment techniques and the insights they yield are only informally coupled to design decisions. (3) Individual risk assessment techniques lack the mix of breadth, fidelity and agility required to span the gamut of the design space.

In this paper we present an approach that addresses these shortcomings. The hallmark of our approach is a simple representation comprising objectives (what the system is to do), risks (whose occurrence would detract from attainment of objectives) and activities (a.k.a. "mitigations") that, if performed, will decrease those risks. These are linked to indicate by how much a risk would detract from attainment of an objective, and by how much an activity would reduce a risk. The simplicity of our representational framework gives it the breadth to encompass the gamut of the design space concerns, the agility to be utilized in even the earliest phases of designs, and the capability to connect to system engineering models and higher-fidelity risk tools.

It is through this integration that we address the shortcomings listed above, and so achieve the intertwining between risk insights and design decisions needed to guide systems engineering towards superior final designs while avoiding costly rework to achieve them. The paper will use an example, constructed to be representative of space mission design, to illustrate our approach.

MODEL-BASED DESIGN

There is a visible movement in the Systems Engineering community to increased use of modeling and simulation. These models are used to evaluate and assess various mission or performance options. These assessments typically involve various parametric, or 'physics-based' models to represent thrust, time delays, telemetry, etc. Because of the dependence on equations, these models rarely even address software, human elements, design flaws, operational work-arounds, degraded operation, quality issues, parts failures or wear-out, among others. Occasionally, they may include some notion of fault-recovery usually at the level of showing that certain paths lead to 'abort' or 'safe mode'. What is needed is more complete system modeling which is agile enough to accommodate the evolutionary nature of the earlier phases of the project life cycle.

In addition, traditional PRA techniques typically focus upon part failures and overall box reliability. The analysts are compelled to develop models which are consistent with the most recently baselined design, while the project design continues to evolve. We refer to this as always working on "the N-1th version". This typically leads to the PRA analyses being performed later in the development life cycle where the design is more static and the data is more plentiful. Unfortunately, this usually does not allow the project to redesign or modify plans to accommodate the insights without great cost and/or schedule impact. What is needed is a way to keep the reliability model more synchronized with the evolving project design, and the ability to handle more than just part failures and end states beyond the usual three: Loss of Mission, Loss of Crew and Loss of Vehicle. In addition, it would be useful to be able to perform more 'triage' earlier in the life cycle, so that even though the data may have more uncertainty, the obvious problems can be brought to light earlier and addressed in a more cost-effective manner.

Meanwhile, real projects still need real gate products for their milestones and for various Reviews and Gates[1]. These required products typically require a lot of effort to generate and even more effort to ensure consistency between the various products. This results in numerous drafts, revisions, and interface meetings. It would be very useful if the technical content which is included in these products could be electronically coupled so that humans could spend most of their time and energy on evaluating their assess adequacy and validity of parameters and assumptions, have tools check for completeness and consistency and then just "press a button" to produce the documents associated with them.

Currently, Configuration Management (CM) is done on products and documents and humans ensure completeness and consistency of these items. Tools and techniques to help in the generation, organization and CM of various objects and their interrelationships (ontologies) are currently on the rise (for example, Vitech Corporation's CORE® product family of engineering development tools, 3SL's Cradle® model based systems engineering environment, and tools based on the SysML Partners "Systems modeling Language (SysML)"). These tools are beginning to make inroads into producing a model-based, self-consistent design which connects Requirements to the Functions which are necessary to achieve them and Functions are connected, in turn, to the Components necessary to perform them. This is a big step and more easily identifies holes, overlaps and inconsistencies between these three principal products.

In parallel, and mostly independently, flight projects generate risk lists, cost profiles and implementation schedules. Many of the costs and schedules are generated phenomenologically, that is, based on historical data for 'similar' missions. It is rare that mitigation plans or options are explicitly included in the schedules or cost summaries. What is needed is a way to more directly connect the cost profiles to the mission implementation approaches which in turn, are connected to the mission risks and associated mitigation plans and options. In addition, the "1:10:100 rule"[2] is alive and well and projects pay the price for items overlooked in earlier design phases. What is needed is a way to explicitly show the relative utility of earlier preventative measures (with more uncertain data and thus more uncertain payoff) versus later mitigation measures with more certain effectiveness data but far greater cost and schedule consequences.

To summarize, it would be very valuable to the flight project community to:

- have more complete, yet agile, system modeling earlier in the life cycle
- be able to synchronize the risk modeling efforts with the rapidly evolving design earlier in the life cycle

[1] Gate Products include items such as Project Implementation Plans, schedules, Risk Management Plans, Launch Approval Plans, Requirements documents, Interface documents, and Mass Equipment Lists among many others.

[2] The 1: 10: 100 rule refers to the idea that if it costs $1 to fix a problem on paper, it costs $10 to fix it in the prototype and $100 to fix it in the flight unit. So, it is more cost effective to eliminate problems early. The trouble is, that it is more difficult to identify the prevention options and their relative benefit in the early phases when specific design data is less available.

- be able to perform earlier identification of driving risks and issues to allow the project team to accommodate changes with far less resource impact.
- have risk evaluations handle more 'shades of grey' in the risk impact assessment, since only a few failures directly result in things like "Loss of Mission". Most failures degrade the mission and it would be valuable to localize and estimate the extent of degradation.
- have requirements, functions, interfaces, schedules and other gate product information electronically coupled within an overall model
- have documents produced automatically from populating templates with mission-specific data.
- have a technique to allow trade-offs between various prevention and mitigation options with evaluation of the consequences not just on mission risk, but also science return, and implementation schedules and costs.

In the sections that follow we first introduce a representative example that will serve as illustration, then describe the advances we have made to allow systems engineering and risk to be intertwined. These advances allow for the representation of representation of risk, and the representation of design and development options. A description of how this all fits together (realized in a software prototype connecting the various tools involved) follows. The paper concludes with a brief description of future plans.

FUNCTIONAL MODEL OF A REPRESENTATIVE EXAMPLE

In order to experiment with our intertwining ideas we developed a prototype representative of the challenges of the problem. We use fragments taken from this prototype as illustration through this paper In this section we summarize its functional model.

Our prototype was intended to explore the challenging edges of the problem and we tuned our functional model in an attempt to produce a tractable, yet interesting model. We loosely followed the current ESMD Lunar mission scenario and included a number of mission phases, each of which required some subset of the available components. We did not include every necessary component in every function, but enough to explore various challenges of generating the complete set of paths through the functional model. It is important to note that since we want to compute the distributions of science return we also require the success paths (in addition to the usual failure paths), since among the paths which return crew and components safely to Earth, different paths can produce more or less science return.

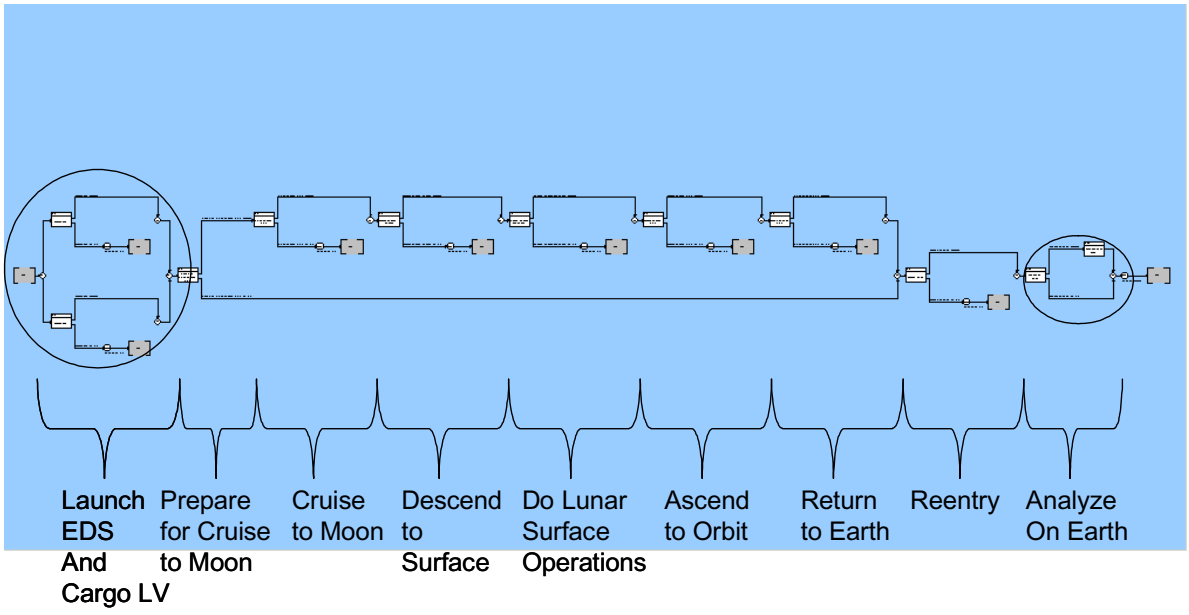


Figure 1 Overall functional flow in the prototype. Note the different phases associated with getting to the Moon, operating on the lunar surface and returning home to Earth. Note also that there are many paths leading to mission termination and one abort path is present (failure exit from Prepare for Cruise to Moon).

Some screenshots of the functional model in our prototype are shown in the following Fig. 1 and 2. These functions are hierarchal in nature and allow users to group high-level functions on one diagram and have detailed functions within each of these, and more details within each of these, etc. We adopted the formalism that functions have at least one success exit and at least one failure exit. The probabilities of the failure exit are the aggregate of the failure probabilities of the constituent components.

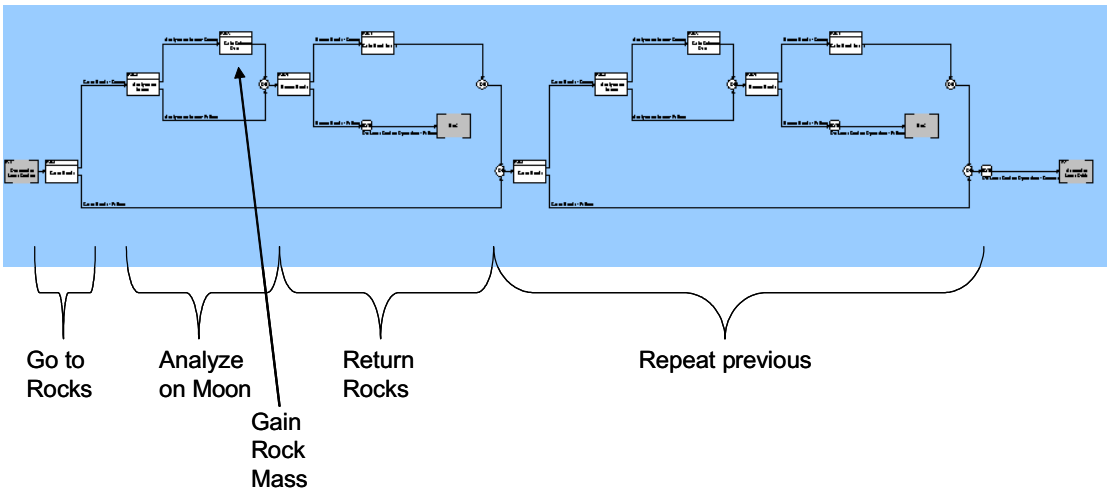


Figure 2 Details of the Do Lunar Surface Operations function. Note that some failures result in Loss of Mission, others result in not acquiring some science resource (either rock mass or science data).

INFUSING RISK INTO MODEL BASED ENGINEERING DESIGN

The many choices among design and development options have ramifications for the reliability of the resulting system. The system's reliability will in turn dictate how successful it is likely to be, as measured by its attainment of the defined mission objectives (in our representative example, these are the measures of Lunar Rock Mass returned to Earth, and Science Data returned to Earth). A wide spectrum of methodologies exist with which to represent and reason about system reliability. Probabilistic Risk Assessment (PRA) is the one we draw upon, described in [1] as "... a comprehensive, structured, and logical analysis method aimed at identifying and assessing risks in complex technological systems for the purpose of cost-effectively improving their safety and performance". We adopt some of the simpler techniques from PRA, but do so in such a way that they are intertwined with systems engineering. The net result is that the reliability model is generated largely automatically from the systems engineering information, rather than requiring separate (manual) construction.

Representation of risk and reliability

Our representation of risk and reliability is hosted within a framework for risk-based decision making that we have been developing and utilizing for several years – the "Defect Detection and Prevention" (DDP) process and accompanying custom software support. The name reflects its origin in the world of quality assurance, where its purpose was to help plan which assurance techniques to use to detect and prevent defects (in spacecraft designs) [2]. It has been applied to the advancement towards spacecraft use of individual technologies, risk assessment for several spacecraft designs, and programmatic decision making for portfolios of multiple spacecraft missions. For descriptions of how it is normally applied, see [3]; for a detailed explanation of the approach, see [4]. This section summarizes the salient aspects for the purposes of this paper.

The DDP framework comprises three kinds of information:

- Objectives – what the system is to accomplish
- Risks – events whose occurrence would detract from the attainment of (some) objectives
- Mitigations – activities that, if applied, will decrease the likelihood and/or severity of risks.

These are linked, indicating for each risk which of the objectives its occurrence would detract from, and by how much, and indicating for each mitigation which of the risks its application would reduce the likelihoods and/or severities, and by how much. The overall set of such information therefore forms a quantitative model that can be

used to compute the overall cost (in terms of resources expended on applying mitigations) and benefit (in terms of sum of expected attainment of objectives). In the basic form of DDP, risks are "atomic" objects, with their individual likelihoods of occurrence.

We recently elaborated DDP's model of risks to allow them to take the form of simple fault trees (composed by using "And", "Or" and "Not" gates) [5]. This elaboration proved to be a key capability in our integration with systems engineering. In this elaborated form, fault tree structures replace DDP's "atomic" risks. The root nodes of those fault trees ("top events" in PRA parlance) are linked to the objectives, indicating by how much occurrence of those top events would detract from attainment of those objectives. Traditional PRA calculations are used to derive the likelihoods of those top events given the structure of the fault trees and the likelihoods of the leaf nodes (the "basic events" in PRA parlance). DDP's mitigations are linked to nodes within the fault trees: the mitigations come in three types:

- "Prevention"-type mitigations (e.g., use of high-quality parts and materials; conducting training of the workforce) are linked to the leaf nodes of fault trees (basic events), indicating by how much they reduce the likelihood of occurrence of those basic events. In DDP, application of a prevention-type mitigation causes those leaf nodes' likelihoods to be reduced accordingly (which in turn will lead to a reduction in the computed likelihood of the failure exit for the top events).
- "Detection"-type mitigations (e.g., tests, analyses, reviews, inspections) are linked to the appropriate level in the fault tree (e.g., a system test is linked to the node at the system level, a component test to the node at the component level), indicating the fraction of the event occurrences at that level in the fault tree that would be identified by application of the mitigation (e.g., a systems test would detect 90% of any latent interface problems within the system that would cause it to fail). There is the assumption that all problems so detected will be corrected, and therefore the net effect will be to increase the reliability of the system as a whole. In DDP, application of a detection-type mitigation causes reductions in the likelihoods of the left nodes within the sub-trees to which the mitigation is connected (which in turn will lead to a reduction in the computed likelihood of the top events)..
- "Alleviation"-type mitigations are linked to the root nodes of fault trees (top events), indicating by how much they reduce the severity (detraction from objectives) of those top events (e.g., backing up critical data will reduce the severity of a system reset). In DDP, application of an alleviation-type mitigation causes reductions in the top-event node's impact on objectives.

Integration with systems engineering

The key to integration of risks within systems engineering is to represent all the possible path through the system engineering model's Functional Flow Block Diagram (FFBD) as a set of mutually exclusive fault trees within DDP. This must be done in such a way that DDP's computation of the likelihood of the top event of the fault tree for a path corresponds to the likelihood of taking that path, and DDP's computation of the attainment of objectives were that fault tree's top event to occur corresponds to the objectives attained if that path is taken. The details are as follows:

The system engineering model's measures of success (in our representative example, these are the measures of Lunar Rock Mass returned to Earth, and Science Data returned to Earth) are represented as DDP objectives. The fault trees representing paths through the FFBD are linked to those objectives so as to indicate how much objective attainment will be had if a path is taken. Since DDP deals with *lack* of attainment (a link in DDP between the root node of fault tree and an objective is used to indicate *detraction* from attainment of that objective should that event occur), as part of our integration we automatically compute the complement of the attainment information as calculated in the systems engineering model, and use that complemented value in the DDP link.

The structure of the DDP fault tree is automatically built to match the path to which it corresponds. A path traverses a sequence of blocks in the FFBD, so the top of the fault tree is an "And" gate, the children of which are each of the blocks traversed by that path. Blocks within the FFBD may be hierarchically decomposed into smaller blocks. The possibility of a failure is represented by a block with two exits: a "success" exit, or a "failure" exit; a specific path through such a block will traverse one of these two exits. Ultimately, the bottommost blocks utilize components in the systems engineering model. Such a block will fail if one or more of the components it utilizes fails to operate correctly. To represent traversal of a failure exit in DDP, a fault tree structure is built that "Or's" together all the components' risks. To represent traversal of a success exit in DDP, the same fault tree as the failure exit is constructed and topped by a "Not" gate (i.e., success occurs if no failure occurs).

Example fragment

The first phase of the overall mission FFBD involves launch of the EDS and launch of the CEV. Failure of either of these leads to aborting the entire mission; only the success of both leads to the next step, preparation for cruise to Moon. Thus in the DDP risk model, there is a risk tree corresponding to launch of the EDS fails, a risk tree corresponding to launch of the EDS succeeds but launch of the CEV fails, and many additional risk trees corresponding to launches of the EDS and CEV both succeeding followed by the various ways in which the remainder of the mission can progress.

In the systems engineering model, EDS utilizes the cargo launch vehicle. Hence the subtree for launch of the EDS fails is an "Or" gate of three children, the three failure modes of that cargo launch vehicle (namely, a design flaw, a parts failure, or an interface problem).

REPRESENTING THE OPTIONS OF MODEL BASED ENGINEERING DESIGNS

The primary purpose of this effort is to encompass within one integrated model both the design and development ramifications (cost, schedule, and other resources needed to build the system) and the operations-time ramifications (expected measures of functionality) *of various options*. In this section we describe how those options, and their ramifications, are represented in our integrated model.

In the systems engineering model, options are represented within the development schedule as "Or" structures that explicitly capture alternative branches of development. We use these to represent a wide variety of choices, encompassing both design alternatives (e.g., whether to use "Class S" [high quality but expensive] parts, or "COTS" parts) and development options (e.g., whether to perform a test at a given stage in development). Also in this same schedule are the mandatory steps that must be done. Steps, whether optional or mandatory, have two kinds of ramifications: *resource* ramifications (how much cost, schedule, and other resources needed to perform the step), and *reliability* ramifications (by how much performing that step will decrease [or, in some instances, increase] a risk or risks). Furthermore, there is a non-trivial coupling between these two, related to the cost of fixing problems discovered in the course of development. For example, if a test reveals a flaw in a unit, then (presumably) that flaw will be corrected. Thus not only does performance of the test itself consume various resources (cost, time), but also the rework/repair/correction that follows from the outcome of the test will also consume resources. The magnitude of that rework effort will depend on:

- how likely it is there are flaws/defects etc. present in the unit at the time of the test (which will be a factor of the design of the unit and the tests, etc., that have previously been performed on it),
- how many resources are needed to rework/repair a given flaw/defect – this will be a depend on both the nature of the flaw/defect, and how late in the development it is discovered. As discussed earlier, the "1:10:100" rule comes into play to escalate the cost of correcting flaws/defects the later in the development they are performed.

We use the systems engineering tool to capture the information necessary to perform these computations, and pass that information over to our risk-based DDP tool to perform them. We adapted DDP to expect schedule structures (including the "Or" branches representing choices, and "And" branches representing parallel activities – important for determining the critical-path(s) of the schedule) as its "mitigations".

The net result is that in DDP it is possible to make a selection of the design and development choices, and compute for that selection both the overall resources consumed, and the expected measures of mission success. Furthermore, DDP helps users explore the tradespace among the options – for example, a cost ceiling can be imposed, and DDP set to use heuristic search to locate a set of design and development choices that achieve near-optimal return of measures of success.

FULLY INTEGRATED MODEL

To address many of the needs identified in the Model-Based Design Section, we have developed an operational prototype which utilizes a major subset of the available schema elements. There are six main elements:

Requirements, Functions, Components, Risks, Work Breakdown Structure, and Implementation

With these six elements (and their interrelationships), we can (in principle) produce the technical content associated with all of the standard Gate Products (e.g. Project Implementation Plan, Risk Management Plan, Mission Operations plan, Cost estimates)

In practice, we have produced partial products in each of these areas.

The real success of the prototype is that we now have explicit coupling between the mission operations side and the project implementation side. For example, when the user chooses to reduce the probability of failure of some

portion of the mission timeline by implementing some risk retirement activity (e.g. longer system test, more peer reviews, better piece parts) not only can they see that the probability of achieving various mission objectives goes up, they can also the additional cost and schedule impacts on the implementation side. Thus, we can produce high reliability, but unaffordable missions and we can also produce risky, but very quick and cheap mission designs and associated plans. Since neither of these extremes is particularly desirable, we actually run a simulated annealing optimization algorithm to find solutions between these two extrema depending on the relative weights assigned to cost, schedule and requirement attainment. Thus, we can save the money associated with reducing the probability of occurrence of a relatively unimportant failure mode and spend it instead to reduce the mission 'tall pole' risks.

We have developed custom software programs for transforming the underlying database including "Integrate Model" which integrates reference and mission data. For example, we have a standard development flow for a typical sub-system, typical system and typical project which we use to populate the development model for the specific mission by finding the components, determining what level it is and populating the Integrated model with an instance of this schedule. Other programs generate the paths, or re-populate the database with user choices or product a Microsoft Project® compatible schedule. Note that this schedule is integrated in the sense that System Integration has as predecessors all of the associated sub-system deliveries. This overall transformation flow is shown in Fig. 3.

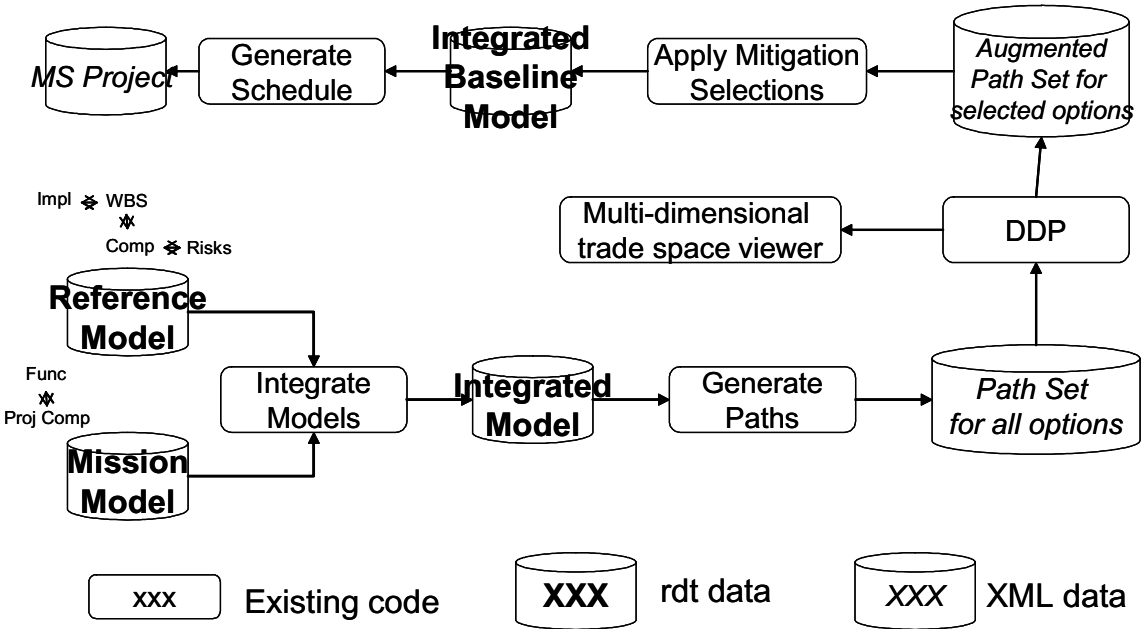


Figure 3 Overall flow of transformation of the database.

Figure 4 shows a representative developmental model, Fig. 5 shows a portion of the complete set of paths through the Functional Model and Fig.6 shows a portion of a resultant schedule.

FUTURE PLANS

Following up on the interest we have generated to date, we intend to increase the complexity of the functional model, increase the fidelity of the cost and effectiveness estimates for the mitigations and expand the scope of the types of failures included in the model. We are well aware of the potential 'exponential explosion' problem in our path generation algorithms, but have as a fall-back the traditional front- and back-end truncation techniques of standard PRA.

However, the primary direction our future plans will take us in application to real cases. We intend to support NASA ESMD design efforts as well as some internal JPL opportunities

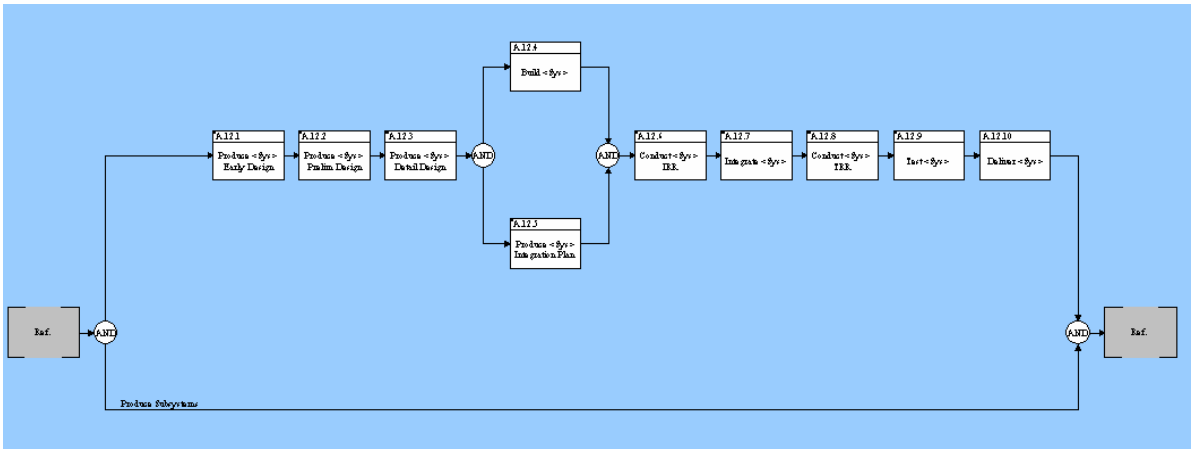
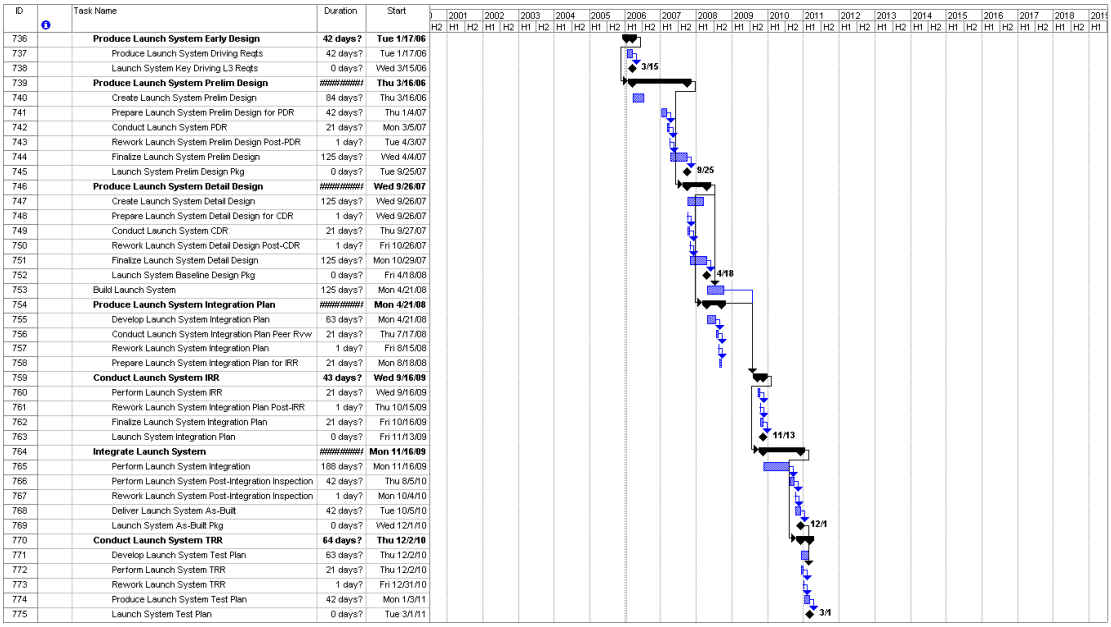


Figure 4 A representative portion of the development flow model for a subsystem

1:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
72:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
87:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
88:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1Failure
102:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1Failure
103:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1Failure
104:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
105:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
106:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
107:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1Failure
108:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1Failure
109:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1Failure
110:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
111:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
112:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
113:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1Failure
114:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1Failure
115:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+6.1.1F-5.1.1Failure
116:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
117:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
118:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
119:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1Failure
120:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1Failure
121:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1Failure
122:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
137:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
152:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
153:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1Failure
167:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1Failure
168:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1Failure
169:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
170:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
171:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
172:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1S-8.2.1Failure
173:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1F-5.1.1S-3.1.1Failure
174:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1F-5.1.1Failure
175:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
176:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
177:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1Failure
178:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1Failure
179:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1S-3.1.1Failure
180:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1S-6.3+6.1.1S-6.5+5.1.1Failure
181:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success
182:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1S-8.2.1S-10Success
183:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-3.1.1S-8.2.1S-8.2.1Failure
184:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1S-3.1.1S-8.2.1Failure
185:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1F-6.1.1S-6.5+6.1.1S-3.1.1Failure
186:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1S-6.2.1F-6.1.1S-6.5+5.1.1Failure
187:1S-2S-3.1.1S-3.2.1S-3.3.1S-3.4.1S-3.5.1S-3.1.1S-4.2.1S-3.3.1S-4.4S-5.1.1S-6.1.1F-6.1.1F-5.1.1S-3.1.1S-8.2.1S-8.2.1S-10S-11Success

Figure 5 Partial set of paths (both failure and success). Note that both failure and success paths can produce non-zero amounts of objective attainment (e.g. acquiring rock mass or science data)



ID	Task Name	Duration	Start
736	**Produce Launch System Early Design**	**42 days?**	**Tue 1/17/06**
737	Produce Launch System Driving Reqts	42 days?	Tue 1/17/06
738	Launch System Key Driving L3 Reqts	0 days?	Wed 3/15/06
739	**Produce Launch System Prelim Design**	**########**	**Thu 3/16/06**
740	Create Launch System Prelim Design	84 days?	Thu 3/16/06
741	Prepare Launch System Prelim Design for PDR	42 days?	Thu 1/4/07
742	Conduct Launch System PDR	21 days?	Mon 3/5/07
743	Rework Launch System Prelim Design Post-PDR	1 day?	Tue 4/3/07
744	Finalize Launch System Prelim Design	125 days?	Wed 4/4/07
745	Launch System Prelim Design Pkg	0 days?	Tue 9/25/07
746	**Produce Launch System Detail Design**	**########**	**Wed 9/26/07**
747	Create Launch System Detail Design	125 days?	Wed 9/26/07
748	Prepare Launch System Detail Design for CDR	1 day?	Wed 9/26/07
749	Conduct Launch System CDR	21 days?	Thu 9/27/07
750	Rework Launch System Detail Design Post-CDR	1 day?	Fri 10/26/07
751	Finalize Launch System Detail Design	125 days?	Mon 10/29/07
752	Launch System Baseline Design Pkg	0 days?	Fri 4/18/08
753	Build Launch System	125 days?	Mon 4/21/08
754	**Produce Launch System Integration Plan**	**########**	**Mon 4/21/08**
755	Develop Launch System Integration Plan	63 days?	Mon 4/21/08
756	Conduct Launch System Integration Plan Peer Rvw	21 days?	Thu 7/17/08
757	Rework Launch System Integration Plan	1 day?	Fri 8/15/08
758	Prepare Launch System Integration Plan for IRR	21 days?	Mon 8/18/08
759	**Conduct Launch System IRR**	**43 days?**	**Wed 9/16/09**
760	Perform Launch System IRR	21 days?	Wed 9/16/09
761	Rework Launch System Integration Plan Post-IRR	1 day?	Thu 10/15/09
762	Finalize Launch System Integration Plan	21 days?	Fri 10/16/09
763	Launch System Integration Plan	0 days?	Fri 11/13/09
764	**Integrate Launch System**	**########**	**Mon 11/16/09**
765	Perform Launch System Integration	188 days?	Mon 11/16/09
766	Perform Launch System Post-Integration Inspection	42 days?	Thu 8/5/10
767	Rework Launch System Post-Integration Inspection	1 day?	Mon 10/4/10
768	Deliver Launch System As-Built	42 days?	Tue 10/5/10
769	Launch System As-Built Pkg	0 days?	Wed 12/1/10
770	**Conduct Launch System TRR**	**64 days?**	**Thu 12/2/10**
771	Develop Launch System Test Plan	63 days?	Thu 12/2/10
772	Perform Launch System TRR	21 days?	Thu 12/2/10
773	Rework Launch System TRR	1 day?	Fri 12/31/10
774	Produce Launch System Test Plan	42 days?	Mon 1/3/11
775	Launch System Test Plan	0 days?	Tue 3/1/11

Figure 6 A portion of a Microsoft Project® schedule produced directly from the prototype model.

ACKNOWLEDGMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through NASA's Exploration Systems Mission Directorate. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

We especially thank Matt Brinza and Vance Heron for their contributions to the software prototypes of these ideas, and Steve Prusha and Steve Wall for their continued support of this effort.

REFERENCES

[1] Stamatelatos, M., Apostolakis, G., Dezfuli, H., Everline, C., Guarro, S., Moieni, P., Mosleh, A., Paulos , T., and Youngblood, R., 2002, "Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners", Version 1.1, prepared for the Office of Safety and Mission Assurance, NASA HQ, Washington, DC, available from http://www.hq.nasa.gov/office/codeq/doctree/praguide.pdf.

[2] Cornford, S.L., "Managing Risk as a Resource using the Defect Detection and Prevention process", *4th International Conference on Probabilistic Safety Assessment and Management,* (September) 1998, New York City, NY, International Association for Probabilistic Safety Assessment and Management.

[3] Feather, M.S., S.L. Cornford, K.A. Hicks & K.R. Johnson, 2005, "Applications of tool support for risk-informed requirements reasoning," *Computer Systems Science and Engineering* **20**(1) (CRL Publishing Ltd) pp. 5-17.

[4] Feather, M.S., and Cornford, S.L., 2003 "Quantitative risk-based requirements reasoning", *Requirements Engineering* (Springer), **8** (4), pp 248-265, 2003; published online 25 February 2003, DOI 10.1007/s00766-002-0160-y.

[5] Feather, M.S., 2004 "Towards a Unified Approach to the Representation of, and Reasoning with, Probabilistic Risk Information about Software and its System Interface", *15th IEEE International Symposium on Software Reliability Engineering*, Saint-Malo, Bretagne, France, 2-5 November 2004, pp 391-402.